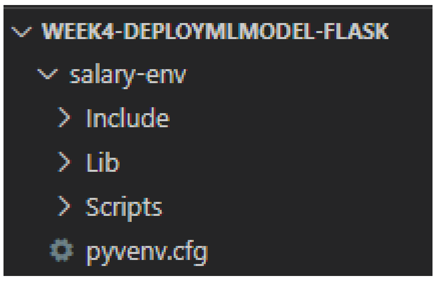
Name: Lauro Cesar Ribeiro

Batch Code: LISP01

Submission date: 15/03/2021

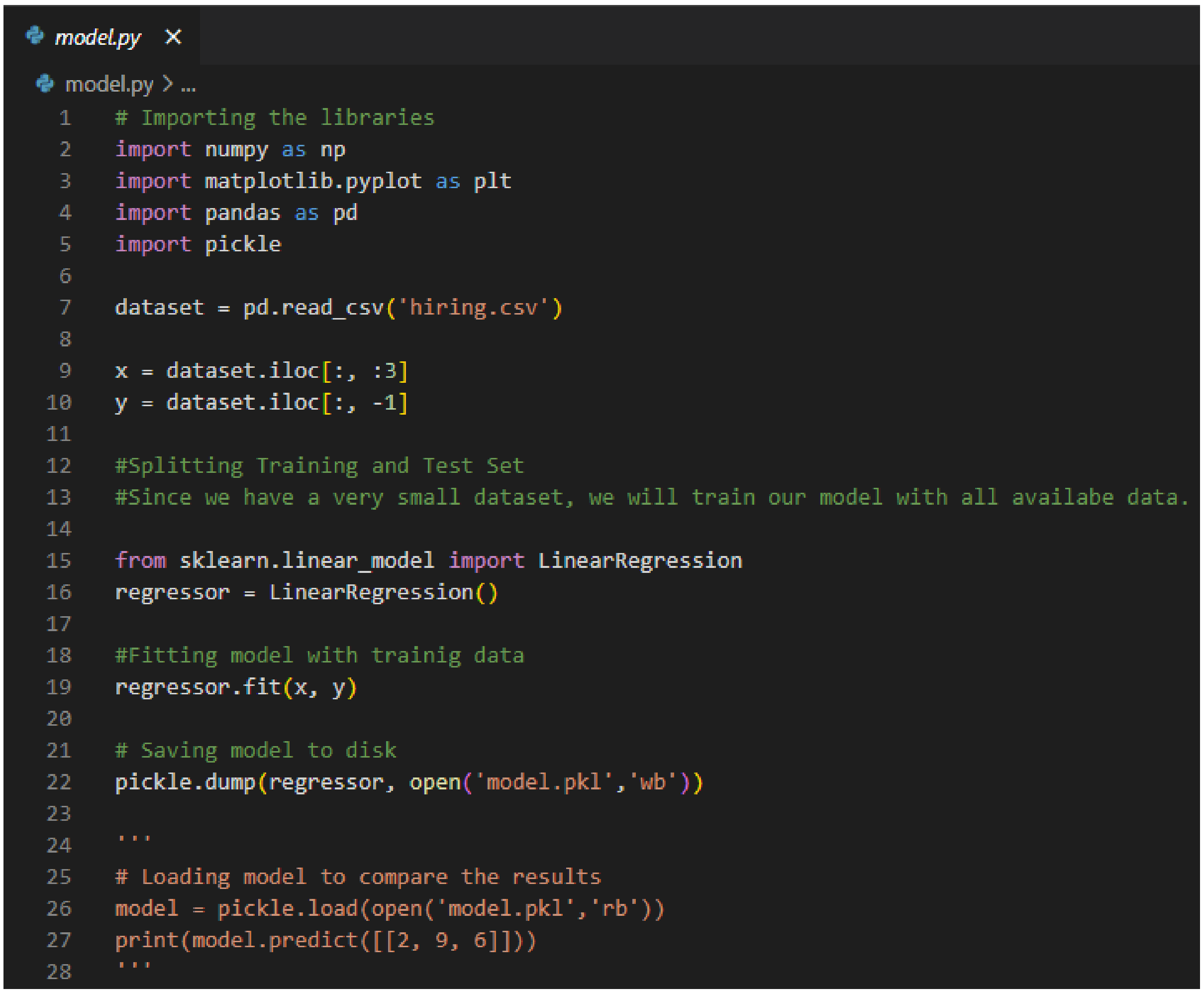
Submitted to: Data Glacier

Snapshots of the Development



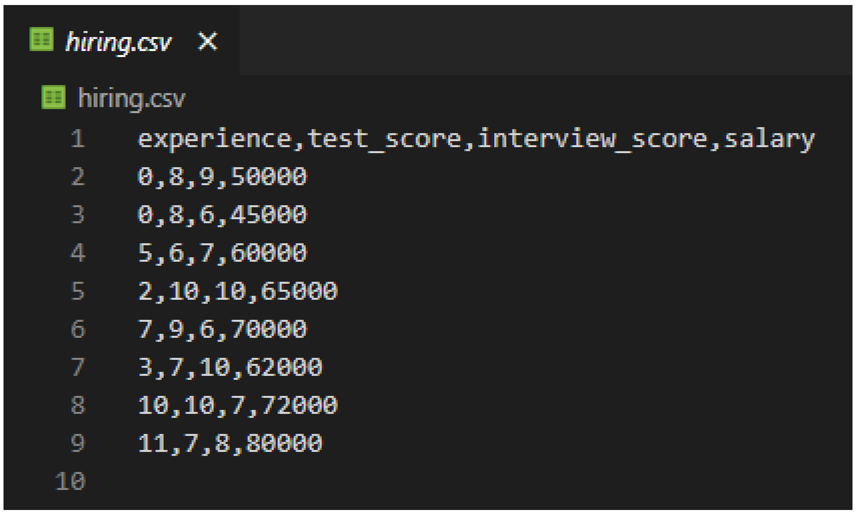
```
WEEK4-DEPLOYMLMODEL-FLASK
  salary-env
    > Include
    > Lib
    > Scripts
      pyvenv.cfg
```

1- Created a Virtual Environment named “salary-env”



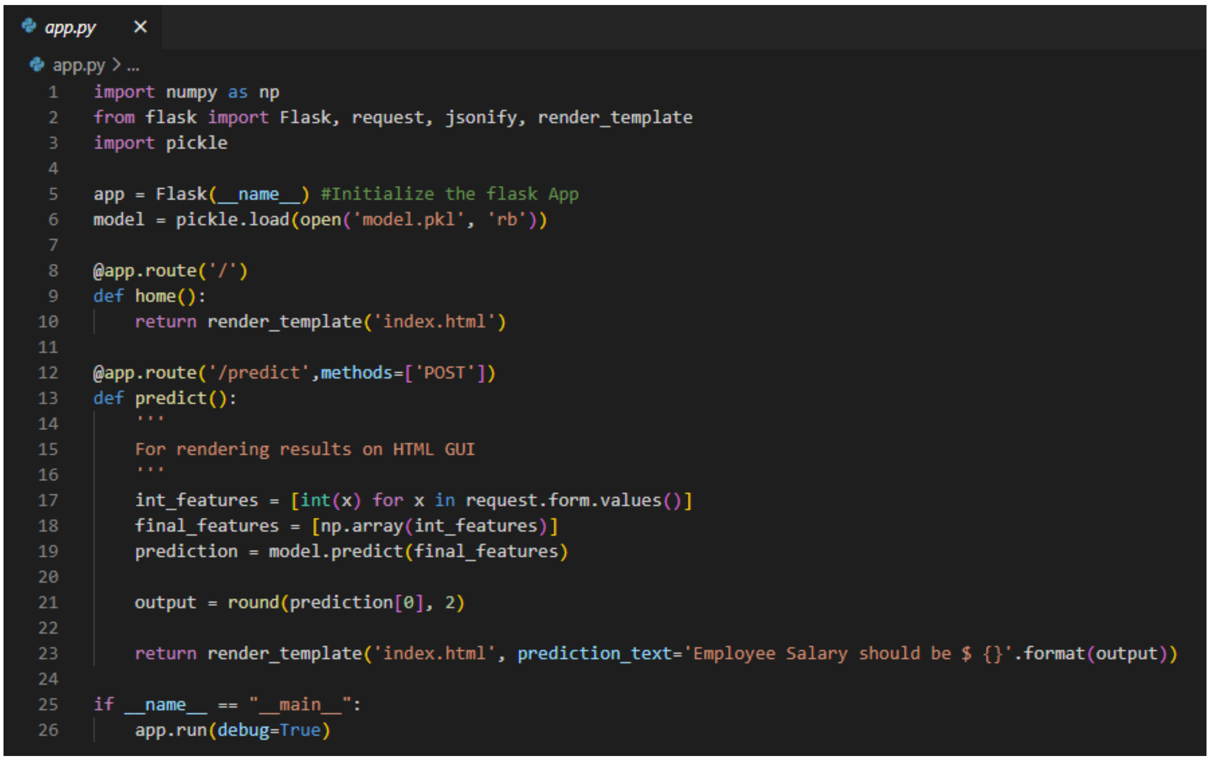
```python
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import pickle

dataset = pd.read_csv('hiring.csv')

x = dataset.iloc[:, :3]
y = dataset.iloc[:, -1]

#Splitting Training and Test Set
#Since we have a very small dataset, we will train our model with all availabe data.

from sklearn.linear_model import LinearRegression
regressor = LinearRegression()

#Fitting model with trainig data
regressor.fit(x, y)

# Saving model to disk
pickle.dump(regressor, open('model.pkl','wb'))

'''
# Loading model to compare the results
model = pickle.load(open('model.pkl','rb'))
print(model.predict([[2, 9, 6]]))
'''
```

2- Trained and Saved my Model.



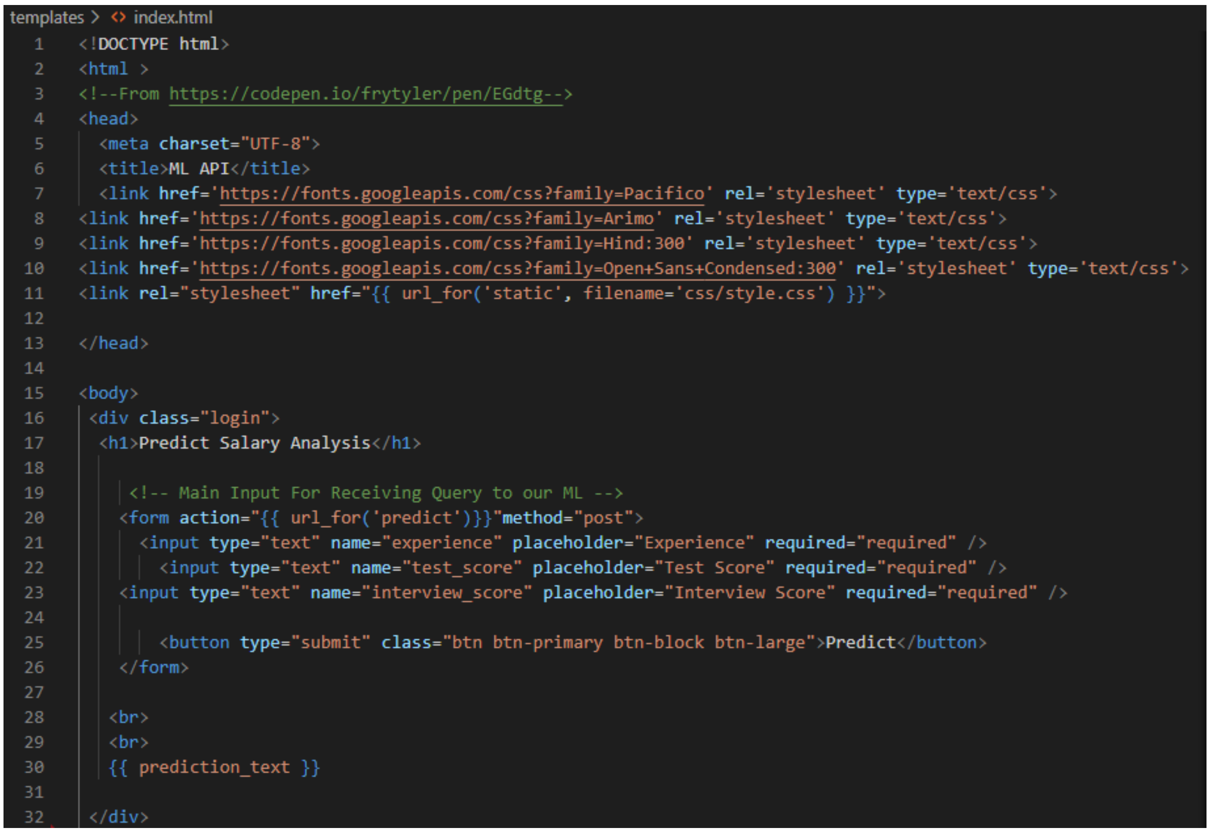
```
experience,test_score,interview_score,salary
0,8,9,50000
0,8,6,45000
5,6,7,60000
2,10,10,65000
7,9,6,70000
3,7,10,62000
10,10,7,72000
11,7,8,80000

```

3- Here is the sample data that I trained my model.



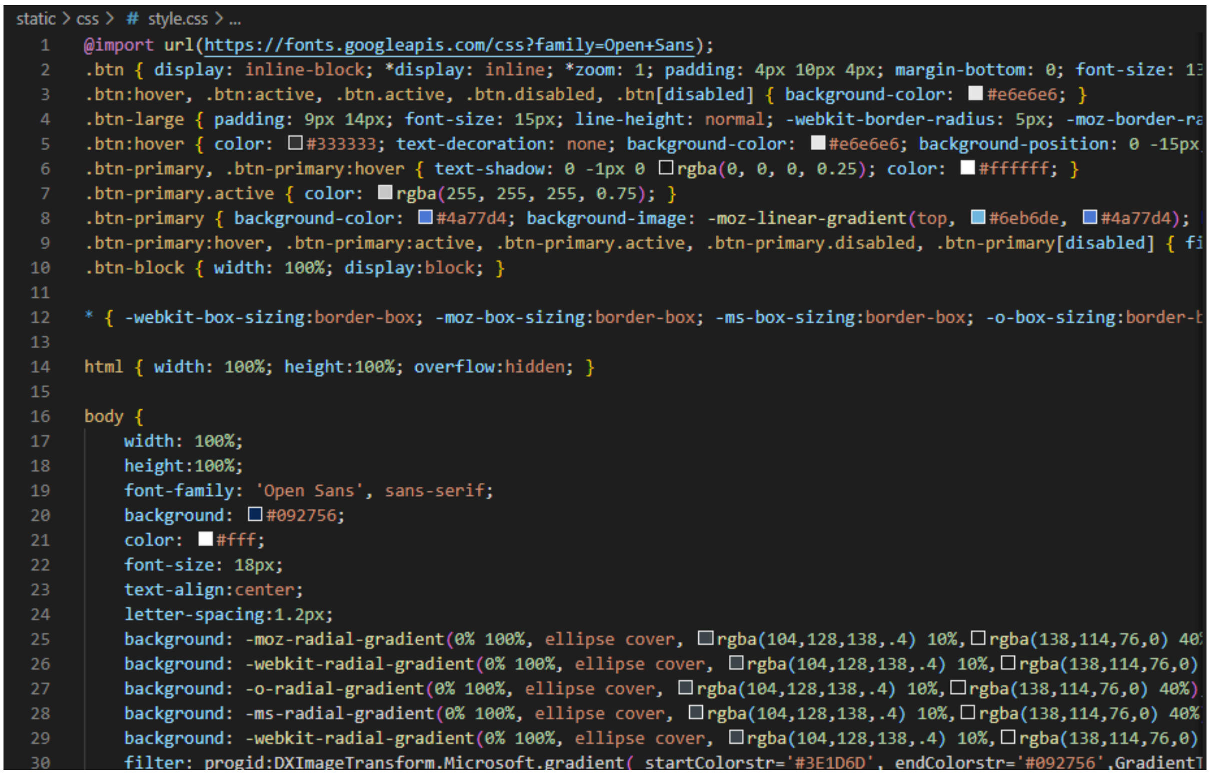
```python
import numpy as np
from flask import Flask, request, jsonify, render_template
import pickle

app = Flask(__name__) #Initialize the flask App
model = pickle.load(open('model.pkl', 'rb'))

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/predict',methods=['POST'])
def predict():
    '''
    For rendering results on HTML GUI
    '''
    int_features = [int(x) for x in request.form.values()]
    final_features = [np.array(int_features)]
    prediction = model.predict(final_features)

    output = round(prediction[0], 2)

    return render_template('index.html', prediction_text='Employee Salary should be $ {}'.format(output))

if __name__ == "__main__":
    app.run(debug=True)
```

4- Developed my Backend using Flask.



```html
<!DOCTYPE html>
<html >
<!--From https://codepen.io/frytyler/pen/EGdtg-->
<head>
  <meta charset="UTF-8">
  <title>ML API</title>
  <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300' rel='stylesheet' type='text/css'>
<link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">

</head>

<body>
 <div class="login">
  <h1>Predict Salary Analysis</h1>

     <!-- Main Input For Receiving Query to our ML -->
    <form action="{{ url_for('predict')}}"method="post">
      <input type="text" name="experience" placeholder="Experience" required="required" />
        <input type="text" name="test_score" placeholder="Test Score" required="required" />
    <input type="text" name="interview_score" placeholder="Interview Score" required="required" />

        <button type="submit" class="btn btn-primary btn-block btn-large">Predict</button>
    </form>

   <br>
   <br>
   {{ prediction_text }}

 </div>
```

5- Inside the folder Templates I created my index page.


static > css > # style.css > ...

```css
@import url(https://fonts.googleapis.com/css?family=Open+Sans);
.btn { display: inline-block; *display: inline; *zoom: 1; padding: 4px 10px 4px; margin-bottom: 0; font-size: 13
.btn:hover, .btn:active, .btn.active, .btn.disabled, .btn[disabled] { background-color: #e6e6e6; }
.btn-large { padding: 9px 14px; font-size: 15px; line-height: normal; -webkit-border-radius: 5px; -moz-border-ra
.btn:hover { color: #333333; text-decoration: none; background-color: #e6e6e6; background-position: 0 -15px
.btn-primary, .btn-primary:hover { text-shadow: 0 -1px 0 rgba(0, 0, 0, 0.25); color: #ffffff; }
.btn-primary.active { color: rgba(255, 255, 255, 0.75); }
.btn-primary { background-color: #4a77d4; background-image: -moz-linear-gradient(top, #6eb6de, #4a77d4);
.btn-primary:hover, .btn-primary:active, .btn-primary.active, .btn-primary.disabled, .btn-primary[disabled] { fi
.btn-block { width: 100%; display:block; }

* { -webkit-box-sizing:border-box; -moz-box-sizing:border-box; -ms-box-sizing:border-box; -o-box-sizing:border-b

html { width: 100%; height:100%; overflow:hidden; }

body {
    width: 100%;
    height:100%;
    font-family: 'Open Sans', sans-serif;
    background: #092756;
    color: #fff;
    font-size: 18px;
    text-align:center;
    letter-spacing:1.2px;
    background: -moz-radial-gradient(0% 100%, ellipse cover, rgba(104,128,138,.4) 10%,rgba(138,114,76,0) 40
    background: -webkit-radial-gradient(0% 100%, ellipse cover, rgba(104,128,138,.4) 10%,rgba(138,114,76,0)
    background: -o-radial-gradient(0% 100%, ellipse cover, rgba(104,128,138,.4) 10%,rgba(138,114,76,0) 40%)
    background: -ms-radial-gradient(0% 100%, ellipse cover, rgba(104,128,138,.4) 10%,rgba(138,114,76,0) 40%
    background: -webkit-radial-gradient(0% 100%, ellipse cover, rgba(104,128,138,.4) 10%,rgba(138,114,76,0)
    filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#3E1D6D', endColorstr='#092756',GradientT
```

6- Inside the Static Folder I created the style.css using Bootstrap to bring my frontend to life.

Final Words:

Project Structure

This project has four major parts:

1. model.py - This contains code for our Machine Learning model to predict employee salaries based on training data in 'hiring.csv' file.

2. app.py - This contains Flask APIs that receives employee details through GUI or API calls, computes the precited value based on our model and returns it.

3. template - This folder contains the HTML template (index.html) to allow user to enter employee detail and displays the predicted employee salary.

4. static - This folder contains the css folder with style.css file which has the styling required for out index.html file.

Running the project

1. Ensure that you are in the project home directory. Create the machine learning model by running below command from command prompt -
```
python model.py
```
This would create a serialized version of our model into a file model.pkl

2. Run app.py using below command to start Flask API
```
python app.py
```
By default, flask will run on port 5000.

3. Navigate to URL http://127.0.0.1:5000/ (or) http://localhost:5000

You should be able to view the homepage.

Enter valid numerical values in all 3 input boxes and hit Predict.

If everything goes well, you should be able to see the predited salary value on the HTML page!
check the output here: http://127.0.0.1:5000/predict